대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0065126
Application Number

출 원 년 월 일 : 2002년 10월 24일
Date of Application OCT 24, 2002

출 원 인 : 한국전자통신연구원
Applicant(s) Electronics and Telecommunications Research Institu

2003 년 05 월 22 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】 특허출원서
【권리구분】 특허
【수신처】 특허청장
【참조번호】 0001
【제출일자】 2002.10.24
【발명의 명칭】 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법
【발명의 영문명칭】 An efficient snapshot technique for shated large storage

【출원인】
【명칭】 한국전자통신연구원
【출원인코드】 3-1998-007763-8
【대리인】
【성명】 권태복
【대리인코드】 9-2001-000347-1
【포괄위임등록번호】 2001-057650-1
【대리인】
【성명】 이화익
【대리인코드】 9-1998-000417-9
【포괄위임등록번호】 1999-021997-1
【발명자】
【성명의 국문표기】 김영호
【성명의 영문표기】 KIM,Young-Ho
【주민등록번호】 731210-1382411
【우편번호】 305-350
【주소】 대전광역시 유성구 가정동 236-1(16/5) 신관 118호
【국적】 KR
【발명자】
【성명의 국문표기】 강동재
【성명의 영문표기】 KANG,Dong-Jae
【주민등록번호】 721213-1067419
【우편번호】 302-243
【주소】 대전광역시 서구 관저동 1144 구봉마을 708동 103호
【국적】 KR

【발명자】
【성명의 국문표기】 박유현
【성명의 영문표기】 BAK,Yu-Hyeon
【주민등록번호】 731123-1024522
【우편번호】 302-766
【주소】 대전광역시 서구 탄방동 산호아파트 102동 409호
【국적】 KR
【발명자】
【성명의 국문표기】 김창수
【성명의 영문표기】 KIM,Chang-Soo
【주민등록번호】 691206-1533317
【우편번호】 305-390
【주소】 대전광역시 유성구 전민동 464-1 엑스포아파트 410동 704호
【국적】 KR
【발명자】
【성명의 국문표기】 신범주
【성명의 영문표기】 SHIN,Bum Joo
【주민등록번호】 571006-1101114
【우편번호】 305-755
【주소】 대전광역시 유성구 어은동 한빛아파트 109동 1402호
【국적】 KR
【발명자】
【성명의 국문표기】 김명준
【성명의 영문표기】 KIM,Myung-Joon
【주민등록번호】 550807-1024619
【우편번호】 305-340
【주소】 대전광역시 유성구 도룡동 393-17번지 23통 3반
【국적】 KR
【심사청구】 청구
【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
권태복 (인) 대리인
이화익 (인)

【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	15	면	15,000 원
【우선권주장료】	0	건	0 원
【심사청구료】	6	항	301,000 원
【합계】	345,000 원		
【감면사유】	정부출연연구기관		
【감면후 수수료】	172,500 원		

【기술이전】

【기술양도】	희망
【실시권 허여】	희망
【기술지도】	희망

【첨부서류】 1. 요약서·명세서(도면)_1통

【요약서】

【요약】

본 발명은 다중 호스트에서 공유되는 대용량 논리 볼륨을 위해 매핑을 기반으로 하는 효율적인 스냅샷 기법에 관한 것이다.

본 발명은 매핑 엔트리에 COW의 수행 여부를 비트로 표현하는 FAB(First Allocation Bit)와 SSB(Snapsho Status Bit)을 도입하여 기존의 스냅샷 기법이 가지는 시간지연 문제들을 해결하였다. 즉, 스냅샷 생성 시 동시에 수행되는 해당 볼륨에 대한 변경 연산의 수행이 스냅샷 생성이 완료될 때까지 지연되는 문제를 해결한다. 또한 스냅샷 생성 후 발생하는 변경 연산에서 COW의 수행 여부를 판단하기 위해 FAB와 SSB를 통하여 스냅샷 매핑 블록을 읽지 않고 원본 매핑 블록만 읽으면 판단이 가능하다. 따라서 스냅샷이 존재하는 볼륨에 대한 쓰기 연산의 수행 시 추가적인 디스크 접근 연산을 줄여서 쓰기 연산의 성능을 향상시킨다. 또한 스냅샷 삭제 수행 시에도 COW가 수행되었는지 판단할 때 스냅샷 매핑 블록에 대한 접근없이 가능하므로 성능 저하를 방지한다. 스냅샷이 1개 이상 존재하는 경우에도 원본 매핑 블록에 대한 접근만으로 판단이 가능하다. 따라서, 스냅샷 개수에 관계없이 항상 일정한 성능을 보장한다.

【대표도】

도 4

【색인어】

스냅샷, 온라인 백업, 매핑 테이블, 대용량 저장장치

【명세서】

【발명의 명칭】

대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법 {An efficient snapshot technique for shated large storage}

【도면의 간단한 설명】

도 1은 일반적인 스냅샷 기법의 Copy-on-Write(COW) 과정을 도시한 도면,

도 2는 본 발명이 적용되는 컴퓨터 시스템의 구성 환경을 도시한 도면,

도 3은 본 발명에 따라 공유 스토리지를 구성하는 논리 볼륨의 계층 구조를 도시한 도면,

도 4는 본 발명에 따른 매핑 테이블 및 매핑 엔트리의 구조를 도시한 도면,

도 5는 본 발명의 일 실시예에 따른 대용량 공유 저장장치를 위한 스냅샷 생성 방법의 동작 흐름도,

도 6은 본 발명의 일 실시예에 따른 대용량 공유 저장장치를 위한 스냅샷 삭제 방법의 동작 흐름도,

도 7은 본 발명의 일 실시예에 따른 대용량 공유 저장장치를 위한 쓰기 연산의 동작 흐름도이다.

도면의 주요부분에 대한 부호의 설명

200: 컴퓨터 시스템 201: SAN 스위치

202: 네트워크 스토리지 풀 203: 호스트 서버

204: LAN 300: 스토리지 풀

310: 디스크 파티션 320: 물리 볼륨

311: 헤더 312: 데이터 영역

313: 볼륨 구성 정보 314: 할당 비트맵 테이블

315: 매핑 테이블 316: 익스텐트

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<16> 본 발명은 대용량 공유 스토리지의 스냅샷 기법에 관한 것으로, 더욱 상세하게는 SAN(Storage Area Network) 기반의 대용량 논리 볼륨을 위해 온라인 백업을 지원하는 스냅샷(Snapshot) 수행방법에 관한 것이다.

<17> 최근들어, 전자 상거래 등 인터넷에 기반을 둔 응용 분야의 기술 발전과, 인터넷 사용자의 확산으로 사용자에게 공유되고, 서비스되는 데이터의 양이 기하 급수적으로 증가하고 있다. 이와 같이 폭발적으로 증가하는 대량의 정보를 효율적으로 공유하고 고속으로 서비스하기 위해서는 대용량의 네트워크 저장장치들이 요구된다.

<18> 대용량의 네트워크 스토리지를 구현하는 기술로는 NAS(Network Attached Storage)와 SAN(Storage Area Network)이 알려져 있는데, SAN은 서버를 거치지 않고 네트워크에 연결된 저장장치를 직접 액세스할 수 있는 데이터 파일 중심의 컴퓨터 시스템 환경을 말한다.

<19> 한편, 24 X 7 X 365 환경을 지원해야 하는 엔터프라이즈 시스템에서는 고속의 대용량 데이터 처리 이외에도 데이터에 대한 가용성 및 신뢰성이 요구된다. 여러 가지 신뢰성 및 가용성을 보장해주는 방법 중 이러한 시스템의 요구 사항을 만족시켜 줄 수 있는 방법으로 온라인 백업의 중요성이 부각되고 있다. 대용량의 데이터에 대한 백업을 수행하기 위해서는 백업 수행 시간이 기하급수적으로 증가하게 된다. 따라서, 백업을 수행하기 위해 시스템을 정지하고 백업이 완료된 후에 다시 서비스를 수행하는 시스템은 사용될 수 없기 때문에, 매핑 테이블 기반의 온라인 스냅샷의 제공이 필수적이다.
여기서, 스냅샷(snapshot)은 사용자가 원하는 특정 시점에서의 데이터 상태를 저장, 유지시켜주는 기법으로 온라인 백업(on-line backup)등에 유용한 기술이다.

<20> 스냅샷 기법은 데이터 전체가 아닌 데이터에 대한 이미지만을 복사해서 스냅샷 생성 시점의 데이터를 그대로 유지하게 된다. 스냅샷 생성 후 데이터 블록에 대한 변경이 발생하면 새로운 블록을 할당해 스냅샷 시점의 데이터를 복사하고 스냅샷의 매핑 엔트리가 새로 할당된 데이터 블록을 매핑하도록 매핑 엔트리 값을 변경한다. 즉, Copy-on-Write(COW)를 수행하여 데이터 블록에 대한 변경이 발생한 뒤에도 스냅샷 시점의 데이터를 유지할 수 있도록 처리한다.

<21> 하지만 기존 매핑 테이블 기반의 온라인 스냅샷 수행에서는 스냅샷 생성 요청이 수행될 때, 스냅샷 매핑 테이블을 복사하는 기간동안 원본 볼륨에 대한 모든 호스트의 접근이 차단되어 서비스를 처리할 수 없다. 대용량으로 볼륨의 크기가 커질수록 매핑 테이블은 커지고, 따라서 입출력(I/O) 접근 제한 시간도 비례해서 증가된다.

<22> 또한 스냅샷 생성 후 데이터 블록에 대한 변경의 처리를 수행하는 Copy-on-Write(COW)와 COW 이후에 발생하는 데이터 블록의 변경은 많은 디스크 I/O를 요

구하게 되어 볼륨에 대한 입출력(I/O) 성능의 저하를 초래한다. 스냅샷 삭제의 수행 시에도 COW에 의해 할당된 데이터 블록의 할당을 위해 원본 볼륨의 데이터 블록과 스냅샷 볼륨의 데이터 블록이 변경되었는지를 검사해서 변경 할당된 데이터 블록을 자유공간 관리자에서 해제해야 하는 과정을 수행하기 때문에 스냅샷 삭제 수행 시간이 길어지게 된다.

【발명이 이루고자 하는 기술적 과제】

<23> 본 발명은 상기와 같은 종래 기술의 문제점을 해결하기 위하여 대용량 공유 스토리지를 제공하는 SAN(Storage Area Network) 기반에서 대용량 논리 볼륨을 위해 온라인 백업을 지원하는 성능이 향상된 스냅샷 수행 방법을 제공하는데 그 목적이 있다.

<24> 즉, 본 발명은 매핑 엔트리에 처음 할당 비트(FAB)와 스냅샷 상태 비트(SSB)의 정보를 추가하여 변경 연산의 수행 및 스냅샷 삭제 시 COW의 판단을 위해 요구되던 스냅샷 매핑 블록에 대한 읽기를 생략하여 변경 연산의 성능을 향상시킨다. 또한 스냅샷 생성 시에도 변경 연산의 동시 수행을 가능하게 하여 대용량의 공유 볼륨 환경에서 성능을 향상시킨 것이다.

<25> 상기 목적을 달성하기 위한 본 발명에 따른 스냅샷 생성 방법은, 스토리지 에리어 네트워크(SAN)기반의 네트워크 스토리지에서 온라인 백업을 위해 스냅샷을 생성하는 방법에 있어서, 매핑 서버가 존재하는 모든 노드의 볼륨 연산 모드를 스냅샷 생성(SNAPSHOT_CREATE) 모드로 변경하는 단계; 매핑 블록의 값을 1씩 증가시키면서 매핑 블록에 대한 잠금을 획득하는 단계; 상기 매핑 블록에 대한 잠금을 획득하지 못하면 복사

완료된 블록의 값을 1씩 증가시키는 단계; 상기 매핑 블록의 잠금을 해제하는 단계; 및 모든 매핑 블록에 대한 복사가 완료되면 스냅샷을 위한 볼륨 구성 정보를 원본 볼륨에서 생성하는 단계를 포함하는 것을 특징으로 한다.

<26> 또한 상기와 같은 목적을 달성하기 위한 본 발명의 스냅샷 삭제 방법은, 스토리지 에리어 네트워크(SAN) 기반의 네트워크 스토리지에서 온라인 백업을 위해 스냅샷을 삭제하는 방법에 있어서, 매핑 서버가 존재하는 모든 노드의 볼륨 연산 모드를 스냅샷 삭제(SNAPSHOT_DESTROY) 모드로 변경하는 단계; 매핑 블록의 값을 1씩 증가시키면서 매핑 블록에 대한 잠금을 획득하는 단계; 매핑 엔트리가 가리키는 데이터 블록이 카피-온-라이트(COW)가 수행되었는지 판단하기 위해 상기 매핑 엔트리의 처음 할당 비트(FAB)와 스냅샷 상태 비트(SSB)를 사용하여 처리하는 단계; 카피-온-라이트(COW)가 수행된 경우 FAB나 SSB를 초기화하고, 상기 매핑 블록의 변경을 디스크에 반영하는 단계; 상기 매핑 블록의 잠금을 해제하는 단계; 및 모든 매핑 블록에 대한 초기화가 완료되면 스냅샷 볼륨을 삭제하는 단계를 포함하는 것을 특징으로 한다.

【발명의 구성 및 작용】

<27> 이하, 본 발명의 바람직한 실시예를 첨부된 도면을 참조하여 설명한다. 본 발명의 실시예에서는 대용량 공유 저장 장치를 위한 스냅샷 기법을 LVM(Logical Volume Manager)에 적용시켜 설명한다. 또한 본 발명에 따른 실시예의 알고리즘을 스냅샷 생성, 스냅샷 삭제, 및 쓰기 연산으로 구분하여 설명하기로 한다.

<28> 도 1은 기존의 일반적인 스냅샷(Snapshot) 기법에서 Copy-on-Write(COW)의 수행 과정을 나타낸 개략도이다.

<29> 도 1을 참조하면, 참조번호 110은 가장 최근의 데이터를 가지고 있는 파일시스템이고, 120은 스냅샷이 생성되었을 때 변경된 데이터에 대해서 스냅샷이 걸린 상태를 유지하기 위해 필요한 데이터를 저장하는 스냅샷 영역이다. 파일시스템(110)에는 파일 A(111)와 파일 B(112)가 저장되어 있고, 파일 A(111)는 A1, A2 블록으로 이루어지고, 파일 B(112)는 B1 내지 B3 블록으로 이루어져 있다.

<30> 이와 같은 구조에서 서버의 응용(application)으로부터 스냅샷이 생성되기 전에 파일 A(111)의 첫번째 블록에 대한 읽기 요청이 들어오면(151), 파일 시스템(110)에 저장되어 있는 파일 A(111)를 읽어서 서비스를 수행한다.

<31> 스냅샷이 생성된 이후에 파일 B(112)의 첫번째 블록에 대한 쓰기가 요청되면(154), 첫번째 갱신인지를 검사하는 과정(140)이 필요하다. 만일, 파일 B(112)의 첫 번째 블록에 대한 쓰기 연산이 첫 갱신이라면 파일 시스템(110)에 있는 파일 B(112)의 첫 번째 블록(B1)을 스냅샷 영역(120)에 복사(B1':122)하고 이의 정보를 변경된 블록맵(121)에 기록한다. 그리고 실제 갱신할 내용은 파일 시스템(110) 내의 파일 B(112)의 첫 번째 블록(B1)에 기록한다. 만일, 처음 갱신이 아니라면 파일시스템(110)의 파일 B(112)의 첫 번째 블록(B1)에 바로 기록한다.

<32> 스냅샷에 대한 읽기 연산에 대해서도 변경된 블록인지를 검사하는 과정(130)이 필요하다. 만일, 변경되지 않은 블록이라면 파일 시스템(110)의 데이터를 바로 읽는다. 하지만 변경되었다고 판단이 되면 스냅샷 영역(120)의 변경된 블록맵(121)을 검사하여 변경된 블록의 물리적인 위치(122)를 찾아 데이터를 읽는다. 즉, 파일 A의 첫번째 블록

(A1)에 대한 읽기(153)라면 변경되지 않은 블록이므로 파일시스템(110)의 A1을 읽어 오고, 파일 B의 첫번째 블록에 대한 읽기(152)라면 변경된 블록이므로 스냅샷 영역의 B1'(122)을 읽어 온다.

<33> 도 2는 본 발명에 따라 논리 볼륨 관리자(LVM: Logical Volume Manager)을 구성하는 컴퓨터 시스템의 물리적인 환경을 나타내는 시스템 구성도이다.

<34> 논리 볼륨 관리자(LVM)는 SAN(Storage Area Network)을 통한 멀티 호스트 환경에서 스토리지 공유를 제공하는 공유 스토리지 클러스터 시스템이다. LVM의 기반이 되는 SAN 환경(200)은 도 2에 도시된 바와 같이 크게 다음의 3가지로 구성되어 있다.

<35> 즉, 데이터를 저장 및 보관하기 위한 네트워크 스토리지 풀(Network Storage Pool: 202)과, 이러한 저장장치(storage device)들을 공유하면서 사용자에게 서비스를 제공하기 위해 이들을 액세스하는 호스트들(203), 그리고 저장장치 (storage device)들과 호스트들(203)을 Fibre Channel(FC)을 통해 연결하고 있는 저장장치 네트워크 연결망인 스위치(Switch)들(201)로 구성된다. 호스트(203)는 스위치(switch: 201)와의 연결을 위해 HBA(Host Bus Adaptor)를 사용하고, 호스트(203)와 스위치(201)의 연결도 Fibre Channel(FC)을 사용하여 구성된다. 각 호스트(Host1~Host4)는 제어 및 관리 정보를 처리하기 위해 LAN(204)으로 연결된다. 이러한 물리적인 환경이 준비되면 논리 볼륨 관리자(LVM)을 사용할 수 있는 환경이 갖추어지게 된다.

<36> 도 3은 본 발명에 따른 컴퓨터 시스템에서 사용되는 대용량 논리 볼륨 관리자를 위한 논리 볼륨(Logical Volume)의 계층 구조를 나타낸다. 논리 볼륨 (Logical Volume)은 SAN을 통해 공유하는 호스트에 제공되는 스토리지 풀(Storage Pool: 300)에서 생성된다. 본 발명에서 제공하는 스토리지 풀(300)은 다음과 같이 3가지로 추상화된다.

<37> 첫번째 추상화는 디스크 파티션(Disk Partition) 또는 물리 파티션(Physical Partition)(310)이다. 디스크 파티션(310)은 운영체제에서 통상적으로 제공되는 툴에 의해 생성되는 데, 디스크 파티션(310)은 논리 볼륨(Logical Volume)의 최소 구성 단위이다.

<38> 즉, 하나 이상의 디스크 파티션(310)이 모여 하나의 논리 볼륨을 형성하는 것이다. 논리 볼륨의 크기 변경 또한 디스크 파티션 단위로 행해지게 된다. 볼륨의 디스크 파티션(310)은 크게 볼륨 헤더 영역(311)과 실제 데이터가 저장되는 데이터 영역(312)으로 구성되는데, 볼륨 헤더 영역(311)은 다시 볼륨의 구성 정보 영역(313)과 할당 비트맵 테이블(314), 및 매핑 테이블(315)로 구성된다.

<39> 두번째 추상화는 물리 볼륨(Physical Volume)(320)이다. 물리 볼륨(320)은 확장 가능한 디스크 파티션들(310)의 집합으로서 이름이 붙여지며, 일련의 연속적인 주소 공간을 이루게 된다. 물리 볼륨(320)의 크기는 시스템 운영 중에 변경 가능하고, 동일한 볼륨 구성 정보(313)를 갖는 디스크 파티션(310)의 모임이다. 물리적인 볼륨(320)을 구성할 물리적인 디스크 파티션(310)에 사용자가 원하는 형태의 볼륨 구성 정보를 갖도록 생성이 된다. 모든 호스트에서 공유하여 사용되고 공유 스토리지 풀(300) 내에서 구별하기 위해 유일한 식별자인 볼륨 식별자(Volume ID)를 갖는다.

<40> 마지막 추상화는 익스텐트(Extent)(316)이다. 익스텐트(316)는 동일 사이즈를 갖고 물리적으로 연속된 블록들의 모임이다. 또한, 정보의 저장을 위해 할당될 수 있는 디스크 공간의 최소 단위이다.

<41> 익스텐트(316)의 크기는 하나의 논리 볼륨에 대해 동일하며, 논리 볼륨 생성 시에 결정되게 된다. 서로 다른 논리 볼륨은 다른 크기의 익스텐트(316)를 가질 수 있고, 그

크기는 2의 지수승으로 물리 디스크(310)의 최소 단위인 하드 섹터 크기의 배수가 되어야 한다.

<42> 디스크 파티션(310)이 운영체제에서 제공하는 툴에 의해 생성된 후 논리 볼륨이 몇 개의 디스크 파티션을 대상으로 정의된다. 이때 볼륨 구성을 위해 필요한 정보가 함께 제공된다. 그러한 정보들은 익스텐트(316)의 크기, 레이드(RAID) 레벨 등을 포함한다.

<43> 이렇게 생성할 볼륨에 대한 명세의 사용자 정의에 따라 논리 볼륨 관리자(LVM)는 해당 물리적인 디스크 파티션(310)에 볼륨 구성에 대한 메타 데이터를 작성하여 기록하게 된다. 볼륨을 구성하는 모든 디스크 파티션의 볼륨 헤더 영역(311)에 구성 정보의 기록이 완료되면 물리적인 볼륨(320)이 생성되고, 생성된 물리 볼륨(320)은 공유하는 모든 호스트에서 등록하여 사용이 가능하다.

<44> 도 4는 본 발명에서 제안하는 스냅샷을 위한 물리 디스크의 구성 및 매핑 테이블 방식을 지원하는 매핑 테이블(315)의 매핑 엔트리(401) 구조를 나타낸다.

<45> 매핑(mapping)은 상위 모듈의 논리 주소를 하위 디스크의 실제 물리 주소로 변환하는 처리를 수행한다. 또한 자유 공간 관리자를 통해 데이터 블록의 효율적인 할당, 해제, 및 논리 주소(402)와 물리 주소(403)간의 독립성을 제공한다. 매핑 테이블(315)을 구성하는 각각의 매핑 엔트리(401)의 구조는 크게 3가지 요소로 구성되어 있다.

<46> 즉, 각 매핑 엔트리(401)는 처음 할당 비트(FAB: First Allocation Bit)(407), 스냅샷 상태 비트(SSB: Snapshot Status Bit)(408), 그리고 물리 디스크 블록의 주소(Physical Address: 403)이다. 물리 디스크 블록의 주소(403)는 다시 디스크 파티션의 주소(Disk_ID: 404)와 물리 블록의 주소 (Physical_Extent_ID: 405)로 구성된다.

<47> 처음 데이터 블록이 자유 공간 관리자로부터 할당되어 사용될 때 매핑 엔트리의 물리 주소(403) 값이 실제 디스크 블록의 위치로 변경된다. 본 발명에 따른 스냅샷 기법에서 매핑 엔트리(401)에 추가한 FAB(407)와 SSB(408)가 기존의 매핑 엔트리 구조와 구별된다.

<48> 처음 할당 비트(FAB: First Allocation Bit)(407)는 모든 매핑 엔트리(401)의 제일 첫 번째 비트를 할당하여 사용한다. FAB(407)는 스냅샷 이후 처음 할당, 사용되는 데이터 블록을 구별하기 위해 사용되는 비트이다. 스냅샷 생성 이후 데이터 블록이 자유 공간 관리자(Free Space Manager)에 의해 할당되고 사용될 때 FAB 값을 '1'로 변경하고, 매핑 엔트리(401)를 디스크에 기록한다.

<49> 스냅샷 상태 비트(SSB: Snapshot Status Bit)(408)는 스냅샷의 상태를 나타내는 비트로서, '1'이면 스냅샷 생성 후 COW가 수행되었다는 의미이고, '0'이면 초기값이나 아직 COW가 수행되지 않은 상태임을 표시한다. SSB(408)는 모든 매핑 엔트리 마다 유지되고 스냅샷 마다 1비트씩 할당되므로 최대 스냅샷 개수 만큼의 비트가 할당된다. FAB(407)와 SSB(408)는 공유 스토리지의 디스크에 물리 볼륨을 생성하는 과정의 매핑 테이블 초기화 과정에서 모두 '0'으로 초기화된다.

<50> 통상, 기존의 스냅샷 기법에서는 스냅샷 이후 처음 사용되는 블록에 대해 그 이후에 발생하는 데이터 변경 연산이 COW를 수행해야 하는지 판단하기 위해 원본 매핑 엔트리 및 스냅샷 매핑 엔트리를 모두 읽어서 물리 주소의 값을 비교해야 한다.

<51> 즉, 두 번의 디스크 I/O를 수행해야 한다. 또한, 스냅샷 삭제 시에 수행되는 COW 수행 블록의 할당 반환의 처리에서 스냅샷 생성 이후 처음 할당된 블록에 대해서는 데이터 블록의 반환 작업을 수행하지 않아야 된다. 이러한 판단을 위해 기존의 스냅샷 기법

에서는 원본 매핑 엔트리와 스냅샷 매핑 엔트리에 대한 두 번의 디스크 I/O를 수행해야 한다.

<52> 그러나 본 발명에 따른 스냅샷 기법에서는 처음 할당 비트(FAB)를 사용하여 오직 원본 매핑 엔트리에 대한 읽기만 수행하면 COW 수행 여부의 판단이 가능하다. 스냅샷 이후 처음 할당 사용되는 데이터 블록에 대해 매핑 엔트리의 제일 처음 비트인 FAB(407)를 '1'로 셋팅하고, 디스크에 매핑 엔트리의 반영을 수행한다.

<53> 첫 번째 문제였던 COW의 수행 여부를 판단하는 과정에서 FAB가 '1'이면, 해당 데이터 블록의 내용을 변경하고 그대로 디스크에 변경을 반영하면 된다. 이미 할당된 블록으로서 FAB가 '0'인 경우에는 스냅샷 상태 비트(SSB) 값을 통해 판단한다.

<54> 스냅샷이 존재하는 경우의 데이터 블록은 각각의 상태에 따라 세 가지 경우로 분류할 수 있다.

<55> 첫 번째는 스냅샷 생성 이후에 처음으로 할당 및 사용되는 데이터 블록이다. 두 번째는 스냅샷 이전에 사용되고 스냅샷 이후에는 아직 변경이 발생하지 않은 데이터 블록이다.

<56> 즉, COW의 수행이 이루어지지 않은 데이터 블록이다. 세 번째는 스냅샷 이후에 변경이 발생한, 즉 COW가 수행된 데이터 블록이다. 스냅샷 생성 이후에 변경되는 데이터 블록의 상태는 위의 3가지 중의 하나이고, 두 번째의 경우에만 변경이 발생하면 COW를 수행해야 한다.

<57> 첫 번째의 경우와 같이 볼륨의 데이터 블록 중 스냅샷 생성 이전에 사용되지 않다가 스냅샷 생성 이후 처음 할당되는 데이터 블록들이 있을 수 있다. 이러한 데이터 블

록에 대한 변경 연산이 발생하면, 처음 변경인 경우 새로운 데이터 블록을 할당하고 디스크에 쓰기를 수행한다.

<58> 그러나 스냅샷 이전에 사용되지 않은 블록이므로 COW는 수행되지 않는다. 즉, COW가 수행되지 않았으므로 SSB의 값이 초기값인 '0'을 그대로 유지하게 된다. 그런데 두 번째 변경부터는 데이터 블록이 사용중이고 스냅샷(SNAPSHOT) 모드에서 SSB값이 '0'이기 때문에 COW를 수행하게 되는 문제점이 있다.

<59> 즉, 본 발명에 따른 스냅샷 기법에서는 스냅샷 매핑 엔트리를 읽어들이기 위한 추가적인 디스크 접근 연산을 하지 않기 때문에 스냅샷 생성 후 처음 사용된 블록과 스냅샷 생성 이전에 할당되고 COW가 수행되지 않은 블록을 구별하는 것이 불가능하다.

<60> 따라서, SSB 이외의 추가 1비트(이것이 FAB이다)를 두어서 스냅샷 생성 후 처음으로 할당되어 사용되는 데이터 블록을 표시하고, FAB에 의해 스냅샷 생성 후 처음 사용된 블록과 스냅샷 생성 이전에 할당되고 COW가 수행되지 않은 블록을 구별한다.

<61> 즉, 스냅샷 생성 이전에 사용되지 않다가 스냅샷 생성 이후 처음 할당되는 데이터 블록의 경우(즉, FAB가 '1'인 경우)에는 SSB가 '0' 이라 하더라도 COW를 수행하지 않는다.

<62> 1. 스냅샷 생성

<63> 스냅샷 생성을 수행하기 위해서는 먼저 스냅샷 생성의 대상이 되는 원본 볼륨에 대한 I/O 및 접근을 차단해야 한다. 이것은 스냅샷 생성 시점의 볼륨 데이터 이미지를 유

지하기 위해서는 스냅샷 생성이 완료될 때까지 원본 볼륨에 대한 변경이 발생하면 안되기 때문이다.

<64> 테이블 기반의 매핑 방법에서는 볼륨에 대한 I/O를 수행하기 위해서는 해당 볼륨에 대한 매핑 테이블이 존재하여야 한다. 즉, 스냅샷 볼륨에 대한 I/O를 수행하기 위해서는 스냅샷 볼륨을 위한 매핑 테이블이 생성되어야 하고, 스냅샷 볼륨에 대한 매핑 테이블 생성 작업을 스냅샷 생성 단계에서 제일먼저 수행한다.

<65> 매핑 테이블의 크기는 볼륨의 크기에 비례하여 커진다. 매핑 테이블의 크기가 커지면 스냅샷 볼륨을 위한 매핑 테이블의 생성 시간이 길어지고, 그만큼 I/O 지연 시간도 늘어나게 된다. 대용량의 공유 스토리지 환경에서는 볼륨의 크기가 수 TB에서 수천 TB 이상 요구되고, 이러한 볼륨에 대한 스냅샷 생성 시간은 몇 십초에서 몇 분 이상까지도 소요된다.

<66> 즉, 이 기간동안 모든 볼륨을 공유 사용하는 호스트의 프로세스 수행이 중단된다. 그러나 몇 십초 이상의 수행 중단은 일반적으로 용인되기 어렵고, 따라서 기존의 스냅샷 생성 방법은 대용량 공유 스토리지 환경에 적합하지 않다.

<67> 본 발명에서는 스냅샷 생성 시 볼륨에 대한 I/O의 지연을 최소화하기 위해 볼륨에 연산 모드 개념을 도입하였다. 볼륨에 대한 연산 모드를 "정상 (NORMAL)" 모드와 "스냅샷 생성 (SNAPSHOT_CREATE)" 모드 그리고 "스냅샷 삭제 (SNAPSHOT_DESTROY)" 모드의 세 가지 모드로 나눈 것이다. 본 발명에 따른 스냅샷 생성 기법은 스냅샷 생성 시 원본 매핑 테이블을 복사해서 스냅샷 매핑 테이블을 생성하는 것은 기존의 스냅샷 기법과 같다.

<68> 그러나 매핑 테이블을 복사할 때, 원본 볼륨에 대한 I/O의 처리를 중단하지 않고 동시에 수행한다. 단지, 호스트에 등록된 볼륨의 연산 모드를 정상 모드에서 스냅샷 생성 모드로 변경하는 동안만 I/O의 지연이 발생하게 된다. 이러한 지연은 매핑 테이블을 복사하는 시간과 비교하면 아주 짧은 시간에 불과하다. 일반 사용자는 거의 인식할 수 없는 무시할 정도의 시간이라고 볼 수 있다.

<69> 도 5는 본 발명에 따라 스냅샷 생성의 흐름을 도시한 흐름도로서, 본 발명에서 수행하는 스냅샷 생성의 흐름을 살펴보면 다음과 같다.

<70> 처음에는 스냅샷 생성으로 변경되는 원본 볼륨에 대한 구성 정보의 변경을 수행한다(501). 스냅샷 개수 등의 정보 변경을 등록된 볼륨에 반영한다. 구성 정보 변경이 완료되면 스냅샷 생성이 요청되면서 완료될 때까지 매핑 서버 호스트의 볼륨의 연산 모드를 "정상(NORMAL)" 모드에서 "스냅샷 생성(SNAPSHOT_CREATE)" 모드로 변경시킨다(502). 스냅샷 생성 모드에서는 매핑 테이블의 복사를 수행하면서, 일반 다른 프로세스가 수행하는 볼륨에 대한 접근 및 I/O의 수행을 동시에 수행한다.

<71> 이어 원본 매핑 테이블의 모든 블록을 스냅샷 매핑 테이블로 복사하기 위해 처음의 매핑 블록부터 모든 블록에 대해 배타 모드의 잠금을 획득(503)하고, 매핑 블록의 복사 작업을 수행한다(504). 잠금을 획득하지 못하면, 모든 매핑 블록의 복사가 완료되었는지 검사한다(506). 잠금을 획득하지 못하는 경우는 동일 블록에 대해 변경을 수행하는 다른 프로세스가 이미 잠금을 획득하고 수행중인 경우이다.

<72> 이 때에는 쓰기 연산에서 볼륨의 연산 모드를 검사해서 COW 작업을 처리해서 해당 매핑 블록에 대한 복사가 수행된다. 잠금을 획득한 경우에는 원본 매핑 블록을 스냅샷 매핑 블록에 복사한다(504). 그리고 매핑 블록에 대한 잠금을 해제한다(505).

<73> 이어 원본 볼륨의 모든 매핑 블록에 대해 복사가 완료되었는지 검사하고(506), 완료되지 않은 경우에는 다음 블록에 대한 잠금 획득(503)한 후 복사 작업을 계속 수행하고, 완료된 경우에는 스냅샷을 위한 볼륨을 할당하고 할당된 볼륨에 원본 볼륨 구성 정보를 복사하고 스냅샷 이름과 스냅샷 순서 등의 정보를 스냅샷 볼륨에 반영한다(507). 스냅샷 볼륨의 생성이 완료되면 매핑 서버 호스트의 연산 모드를 "정상" 모드로 변경시킨다(508). 이러한 방법으로 스냅샷 생성을 수행하면 원본 볼륨을 접근하는 응용 프로그램의 I/O 중단없이 스냅샷 생성을 정상적으로 수행할 수 있다.

<74> 2. 스냅샷 삭제

<75> 한편, 기존의 스냅샷 기법에서 성능 저하를 초래하는 요인 중의 하나가 스냅샷 삭제 연산 수행 시의 COW의 수행 여부를 판단하기 위해 스냅샷 매핑 블록을 접근하는 오버헤드로 인한 성능 저하이다. 스냅샷 삭제 연산에서 오버헤드를 유발하는 부분은 COW의 수행으로 할당된 데이터 블록에 대한 할당을 해제하는 작업이다.

<76> 즉, 기존의 스냅샷 기법에서는 COW가 수행되었는지 여부를 판단하기 위해 원본 매핑 엔트리와 스냅샷 매핑 엔트리를 모두 읽어서 두 엔트리가 가리키는 물리 블록의 주소가 일치하는가 확인한다. 두 물리 블록의 주소가 일치하면, COW가 수행되지 않은 데이터 블록으로 매핑 엔트리의 주소만 초기화시키면 된다.

<77> 일치하지 않으면, 스냅샷 매핑 엔트리가 지시하는 데이터 블록에 대한 할당을 해제하는 작업을 수행한 후 매핑 엔트리의 초기화를 수행한다. 스냅샷이 한 개 이상일 경우

에는 다른 스냅샷의 매핑 엔트리까지 비교해야 한다. 즉, 최소한 두 번 이상의 추가적인 디스크 I/O 작업을 수행해야 한다.

<78> 이와 같이 스냅샷 삭제 시 블록 할당 해제 여부를 판단하기 위해 스냅샷 매핑 블록에 대한 접근으로 성능 저하가 발생되는 기존 스냅샷 삭제 연산의 문제점을 해결하기 위해 본 발명에서는 스냅샷 상태 비트(SSB: 408)와 처음 할당 비트(FAB: 407)를 원본 매핑 엔트리에 도입하여 해결하였다. SSB(408)는 물리 볼륨이 생성되는 과정에서 매핑 블록이 초기화될 때 '0'으로 초기화되고, COW가 수행될 때 '1'로 변경된다.

<79> 본 발명에 따른 스냅샷 기법에서는 스냅샷 매핑 엔트리를 읽어서 원본 매핑 엔트리와 비교하는 방법을 사용하지 않고, 원본 볼륨의 매핑 엔트리만 읽으면 COW의 수행 여부를 판단할 수 있다. 즉, 제일 먼저 FAB(407) 값을 검사하고 FAB(407)가 '1'이면 삭제가 수행되는 스냅샷이 첫 번째 스냅샷인지 확인한다. FAB가 '1'이고 첫 번째 스냅샷인 경우, 해당 데이터 블록은 스냅샷이 생성된 후 처음으로 할당, 사용된 블록이므로 할당 해제를 수행하지 않는다. 그 외의 경우에는 SSB 값을 검사한다.

<80> 삭제하는 스냅샷 위치의 SSB(408)가 '0'이면, COW가 수행되지 않은 경우이고, SSB(408)가 '1'이면 COW가 수행된 경우이다. SSB(408)가 '0'이면 다음 블록에 대한 수행을 계속하고, SSB(408)가 '1'인 경우에는 다음 스냅샷이 존재하는가와 스냅샷이 존재하는 경우, 그 스냅샷의 SSB 값을 확인한다.

<81> 다음 스냅샷이 존재하지 않거나 스냅샷이 존재하는데 COW가 수행된 경우에는 현재 데이터 블록의 할당을 해제해야 한다. 이러한 경우는 다음 스냅샷이 존재하는 경우, SSB 값을 통해 COW의 수행을 판단하고, 현재 스냅샷에 대한 할당의 해제 여부를 결정한

다. 즉, 본 발명에 따른 스냅샷 기법에서는 스냅샷 매핑 엔트리 값을 읽지 않고 원본 매핑 엔트리를 통해 스냅샷 삭제 작업의 수행이 가능하다.

<82> 스냅샷의 개수가 여럿인 경우, 기존의 스냅샷 기법에서는 스냅샷 개수만큼의 매핑 엔트리를 읽는 I/O 작업을 수행해야 된다. 현재 스냅샷 엔트리 외에 다음 스냅샷의 매핑 엔트리를 비교하여 작업을 수행한다. 그러나 본 발명에 따른 방법은 스냅샷의 개수에 상관없이 원본 매핑 엔트리 하나만 읽으면 모든 처리를 수행할 수 있다. 따라서, 스냅샷이 많을수록 본 발명에 따른 스냅샷 기법의 삭제 연산의 성능이 좋아진다.

<83> 도 6은 본 발명에 따라 스냅샷 삭제의 흐름을 도시한 흐름도로서, 본 발명의 스냅샷 삭제의 흐름을 살펴보면 다음과 같다.

<84> 스냅샷 삭제의 수행은 스냅샷 생성 후 발생한 데이터 블록의 변경으로 발생한 COW에 의해 할당된 데이터 블록의 할당을 해제하고, 스냅샷을 위한 매핑 테이블을 삭제한 후 스냅샷 볼륨을 삭제한다.

<85> 스냅샷 삭제가 요청되면 매핑 서버의 볼륨 연산 모드를 "스냅샷 삭제 (SNAPSHOT_DESTROY)" 모드로 변경한다(601). 스냅샷 삭제 모드로 변경하는 이유는 스냅샷 삭제가 완료되기 전에 발생하는 데이터 블록에 대해 해당 스냅샷에 대한 COW가 수행되는 것을 방지하기 위해서이다. 데이터 블록의 할당 해제 여부를 판단하기 위해 원본 매핑 테이블 엔트리의 FAB(407)와 SSB(408)값을 검사해야 한다. 따라서, 데이터 블록을 접근하기 위한 매핑 엔트리를 읽기 위해 원본 볼륨 매핑 엔트리가 저장된 디스크 블록의 위치를 구한다(602). 디스크 블록에서 매핑 블록을 메모리로 읽어서 매핑 엔트리를 얻는다(603). 매핑 블록의 모든 엔트리에 대해 차례로 COW의 수행을 검사하는 작업을 수행한다.

<86> 이어 매핑 엔트리의 FAB와 SSB값을 통해 COW가 수행되었는지 판단한다(604). COW가 수행되지 않은 블록이면 다음 매핑 엔트리에 대해 검사하는 작업(609)을 수행한다. COW가 수행된 데이터 블록이면, 데이터 블록의 할당을 해제할 것인지 판단하는 작업을 수행한다(606). COW가 수행된 데이터 블록의 할당을 해제하는 경우는 두 가지이다. 첫째로 다음 스냅샷이 존재하지 않는 경우와, 둘째로 다음 스냅샷이 존재할 때 다음 스냅샷 생성 후 동일 데이터 블록에 COW가 발생한 경우이다. 위의 두 가지 경우에 할당된 데이터 블록을 해제하고(607), 매핑 엔트리의 현재 스냅샷에 대한 상태 비트 값인 SSB를 '0'으로 초기화한다(608). 데이터 블록을 해제하지 않는 경우에는 바로 SSB 값만 '0'으로 초기화한다. 여기까지 수행하면 하나의 매핑 엔트리에 대한 수행이 완료된다.

<87> 매핑 블록에 존재하는 모든 엔트리에 대해 수행이 완료되었는지 검사한다(609). 완료되지 않은 경우에는 다음 매핑 엔트리에 대해 단계 605에서 단계 608까지의 작업을 수행한다. 매핑 블록의 모든 엔트리에 대해 수행이 완료된 경우, COW가 한 번 이상 발생했으면 매핑 블록을 디스크에 반영하는 작업을 수행한다(610).

<88> 매핑 블록에 대한 쓰기 작업이 완료되면, 매핑 블록에 대한 잠금을 해제한다(611). 잠금 해제 후에는 모든 매핑 블록에 대해 수행이 완료되었는지 검사한다(612). 수행할 매핑 블록이 존재하는 경우에는 단계 602에서 다음 매핑 블록을 구하고, 단계 611까지의 과정을 수행한다.

<89> 모든 매핑 블록에 대해 수행한 경우에는 스냅샷 볼륨을 호스트에서 삭제하는 작업을 수행한다(613). 스냅샷 볼륨을 삭제하면 실제적인 스냅샷 삭제 수행이 완료되고, 스냅샷 삭제로 변경되는 원본 볼륨의 구성 정보의 반영을 수행한다(614). 마지막으로 매

핑 서버에 존재하는 볼륨의 연산 모드를 정상 모드로 변환하면(615), 스냅샷 삭제 과정이 완료된다.

<90> 3. 데이터 쓰기 연산

<91> 스냅샷의 성능을 평가 결정하는 중요한 요소 중의 하나가 스냅샷 생성 후 데이터 블록에 변경이 발생했을 때 데이터 블록의 변경을 반영하는 연산이 얼마나 효율적인지로 평가된다. 스냅샷 모드의 읽기 연산은 일반 정상 모드의 읽기 연산과 동일하게 처리된다. 매핑을 통해 논리 블록과 일치하는 물리 데이터 블록을 구하고, 얻어진 물리 데이터 블록에서 데이터를 읽어오는 과정으로 수행된다.

<92> 스냅샷이 존재할 때 성능 저하를 초래하는 연산이 데이터 블록의 변경을 반영하는 쓰기 연산이다. 매핑 테이블을 기반으로 하는 스냅샷 기법에서는 두 가지 경우의 쓰기 연산이 존재한다.

<93> 첫 번째는 스냅샷 생성 이전에는 할당 사용되지 않다가 스냅샷 생성 이후에 새로이 할당되어 쓰여지는 데이터 블록이다. 스냅샷 볼륨은 스냅샷이 생성되는 순간의 볼륨 데이터 이미지만 유지하므로 스냅샷 생성 이후에 쓰여진 데이터와는 무관하다.

<94> 따라서, 스냅샷 생성 이후 쓰여지는 데이터는 다른 처리 과정없이 정상 모드의 쓰기 연산과 동일하게 자유 공간 관리자로부터 데이터 블록을 할당받고 원본 볼륨의 매핑 엔트리에 데이터 블록의 물리 주소를 반영한 후 디스크 블록에 내용 변경의 쓰기 작업을 수행한다.

<95> 두 번째는 스냅샷 생성 이전에 쓰여지고 생성 이후 변경이 발생한 데이터이다. 스냅샷은 생성 시점의 볼륨 데이터를 그대로 유지해야 된다. 따라서 스냅샷 생성 이전에 쓰여진 데이터의 내용은 변경이 발생해도 그대로 유지되어야 한다. 이렇게 스냅샷 이전에 할당된 데이터 블록 내용의 유지를 위해 수행하는 작업을 Copy-on-Write(COW)라 한다.

<96> 이러한 COW의 과정은 매핑 테이블을 기반으로 하는 스냅샷 기법에서는 모두 동일하게 수행해야 하는 과정이다. 문제는 COW가 수행된 후 발생되는 데이터 블록의 변경에 대한 처리이다. 기존의 스냅샷 기법에서는 변경을 반영하려는 데이터 블록이 COW가 수행되었는지를 판단하기 위해 원본 매핑 엔트리와 스냅샷 매핑 엔트리를 모두 읽어서 두 엔트리가 매핑하는 물리 블록의 주소가 동일한가를 비교하여 COW의 수행 여부를 판단하였다.

<97> 즉, 스냅샷 매핑 블록에 대한 I/O가 추가적으로 필요하다. 이러한 과정은 스냅샷의 개수가 증가하면 추가로 요청되는 디스크 I/O 횟수도 스냅샷의 수에 비례하여 증가한다. 즉, 스냅샷이 두 개이면 두 번의 I/O가, 세 개이면 세 번의 I/O가 더 수행되어야 한다. 이렇게 되면 존재하는 스냅샷의 수에 비례하여 쓰기 연산의 성능 저하가 더욱 심해진다.

<98> 본 발명에서는 다음과 같은 방법으로 기존의 스냅샷 기법이 갖고 있는 데이터 쓰기 연산의 문제를 해결하였다. SSB는 볼륨의 모드가 스냅샷 모드이고 스냅샷 이전에 할당, 사용되는 데이터 블록에 대해 스냅샷 생성 이후 데이터 블록의 내용 변경이 처음 발생하면 '1'로 변경된다.

<99> 즉, COW가 수행되는 매핑 엔트리의 스냅샷에 해당하는 SSB의 값이 '1'로 변경된다. COW가 수행된 이후의 동일 데이터 블록에 대한 변경이 발생하면 원본 볼륨의 매핑 엔트리의 SSB 값을 통해 COW의 판단을 처리한다. 즉, 본 발명에 따른 스냅샷 기법에서는 스냅샷 매핑 엔트리를 읽어서 원본 매핑 엔트리와 비교하여 판단하는 작업을 수행하지 않아도 원본 볼륨 매핑 엔트리를 통해서만 수행을 할 수 있다.

<100> 스냅샷의 개수가 많아지면 성능은 더욱 좋아지게 된다. COW가 수행된 이후 발생하는 데이터 블록의 변경으로 인한 쓰기 연산은 정상 모드의 쓰기 연산과 거의 같은 성능을 갖는다.

<101> 도 7은 본 발명에 따른 시스템에서 스냅샷이 존재할 때 사용자의 볼륨 디스크에 대한 I/O 요청을 처리하는 쓰기 연산의 흐름도를 나타낸다. 볼륨에 대해 I/O 요청이 발생하면, 데이터 블록에 대한 매핑 정보가 저장된 매핑 블록의 물리적인 디스크 및 블록의 주소를 구한다(701).

<102> 매핑 블록에 대한 배타 모드의 잠금을 획득하고(702), 매핑 블록을 디스크에서 메모리의 버퍼로 읽어오고 논리 주소에 해당되는 매핑 엔트리를 얻는다(703). 볼륨의 연산 모드를 검사한다(704). 즉, 현재의 I/O가 스냅샷 생성이나 삭제 중에 발생한 것인지를 검사한다.

<103> 볼륨의 연산 모드가 정상 모드(NORMAL)이면 볼륨에 스냅샷이 존재하는가 검사한다(705). 볼륨에 스냅샷이 존재하지 않는 경우에는 일반 쓰기 연산처럼 데이터 블록을 디스크에 기록하고(716), 매핑 블록의 잠금을 해제하고 종료한다(717). 볼륨에 대한 스냅샷이 존재하는 경우에는 데이터 블록이 스냅샷 생성 이전에 사용되지 않았는지를 검사한

다(707). 스냅샷 생성 이후 할당, 사용되는 데이터 블럭에 대해서는 COW가 수행되지 않는다.

<104> 따라서, 변경된 내용을 데이터 블록의 디스크에 기록하고(716), 매핑 블록에 대한 잠금을 해제하고(717) 종료한다. 데이터 블록이 스냅샷 생성 이전에 할당된 경우에는, 스냅샷 생성 이후 동일 데이터 블록에 대한 변경으로 COW가 이미 수행되었는지 검사한다(708).

<105> COW가 이미 수행된 경우에는 데이터 블록을 디스크에 기록하고(716), 매핑 블록에 대한 잠금을 해제하고 수행을 종료한다(717). COW가 아직 수행되지 않은 경우에는 COW를 수행해야 한다. 원본 매핑 블록과 동일 논리 주소의 스냅샷 매핑 블록을 버퍼로 읽어서 스냅샷 매핑 엔트리를 얻고(709), COW를 수행하기 위해 새로운 물리 데이터 블록을 할당받은(710) 다음, 데이터 블록의 내용을 새로 할당받은 데이터 블록에 복사하고 복사된 데이터 블록을 볼륨 디스크에 기록한다(711).

<106> 스냅샷 매핑 엔트리가 가리키는 물리 주소를 새로 할당받은 데이터 블록의 주소로 변경하고(712), COW가 수행되었음을 나타내는 원본 매핑 엔트리의 현재 스냅샷에 대한 SSB값을 1로 변경하고(713), 스냅샷 매핑 블럭을 디스크에 기록한다(714). 그리고 원본 매핑 블록을 디스크에 기록한다(715). 단계 709에서 단계 715까지의 COW 과정을 수행하면 데이터 블록의 내용을 디스크에 기록하고(716), 매핑 블록에 대한 잠금을 종료하고 수행을 종료한다(717).

<107> 단계 704에서 볼륨의 연산 모드가 정상 모드가 아니고 스냅샷 생성 모드이면 매핑 엔트리를 포함하는 매핑 데이터 블록에 대한 복사가 수행되었는지 검사한다(706). 복사가 완료된 경우에는 COW가 수행되었는지 판단하기 위해 SSB값을 검사한다. SSB값이 '1'

이면 이미 COW가 수행된 경우로 데이터 블록을 디스크에 기록하고(716), 매핑 블록에 대한 잠금을 해제하고(717) 수행을 종료한다.

<108> SSB값이 '0'이면 COW 작업을 수행해야 한다. 단계 709에서 단계 715까지의 COW 작업을 수행하고, 데이터 블록을 디스크에 기록하고(716), 매핑 블록의 잠금을 해제하고(717) 수행을 종료한다. 만약, 복사가 완료되지 않았다면 COW가 수행되어야 한다. 단계 709에서 단계 717까지의 작업을 수행하고 종료한다. 스냅샷 매핑 블록은 COW가 수행되면 자동적으로 복사가 수행된다.

【발명의 효과】

<109> 상술한 바와 같이 본 발명에 따른 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법은 대용량 공유 스토리지 환경에서 스냅샷 생성 과정에서 응용 프로그램의 동시 수행을 가능하게 하고, 스냅샷 생성 후에 발생하는 변경 연산의 성능을 향상시켜 24*7*365의 고 가용성이 요구되는 웹 서버나 전자 상거래와 같은 엔터프라이즈 시스템에서 데이터의 가용성, 신뢰성을 보장하고 성능 저하가 발생하지 않으면서 온라인 백업을 지원할 수 있는 장점이 있다.

<110> 이상에서 설명한 것은 본 발명에 따른 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법을 설명한 하나의 실시 예에 불과한 것으로써, 본 발명은 상기한 실시 예에 한정되지 않고, 이하의 특허 청구의 범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변경 실시가 가능한 범위까지 본 발명의 기술적 사상이 미친다고 할 것이다.

【특허청구범위】

【청구항 1】

다수의 호스트들이 대용량의 네트워크 저장장치를 공유하도록 된 스토리지 에리어 네트워크(SAN) 기반의 시스템에 있어서,

상기 네트워크 저장장치에 논리 볼륨을 구성하는 디스크의 레이 아웃에서 원본 매핑 테이블과 스냅샷 매핑 테이블을 동일 디스크에 저장하고, 스냅샷 볼륨이 추가되어 카피-온-라이트(COW)가 발생하면 동일 디스크 상에서 데이터 블록을 할당하는 것을 특징으로 하는 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법.

【청구항 2】

제 1 항에 있어서, 상기 원본 매핑 테이블의 매핑 엔트리는

물리 데이터 블록 정보와, 스냅샷 상태 정보를 포함하는 것을 특징으로 하는 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법.

【청구항 3】

제 2 항에 있어서, 상기 매핑 엔트리의 스냅샷 상태 정보는

스냅샷 상태 비트(SSB: Snapshot Status Bit)와 처음 할당 비트(FAB: First Allocation Bit)인 것을 특징으로 하는 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법.

【청구항 4】

스토리지 에리어 네트워크(SAN)기반의 네트워크 스토리지에서 온라인 백업을 위해 스냅샷을 생성하는 방법에 있어서,

매핑 서버가 존재하는 모든 노드의 볼륨 연산 모드를 스냅샷 생성 (SNAPSHOT_CREATE) 모드로 변경하는 단계;

매핑 블록의 값을 1씩 증가시키면서 매핑 블록에 대한 잠금을 획득하는 단계;

상기 매핑 블록에 대한 잠금을 획득하지 못하면 복사 완료된 블록의 값을 1씩 증가시키는 단계;

상기 매핑 블록의 잠금을 해제하는 단계; 및

모든 매핑 블록에 대한 복사가 완료되면 스냅샷을 위한 볼륨 구성 정보를 원본 볼륨에서 생성하는 단계를 포함하는 것을 특징으로 하는 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법.

【청구항 5】

스토리지 에리어 네트워크(SAN) 기반의 네트워크 스토리지에서 온라인 백업을 위해 스냅샷을 삭제하는 방법에 있어서,

매핑 서버가 존재하는 모든 노드의 볼륨 연산 모드를 스냅샷 삭제 (SNAPSHOT_DESTROY) 모드로 변경하는 단계;

매핑 블록의 값을 1씩 증가시키면서 매핑 블록에 대한 잠금을 획득하는 단계;

매핑 엔트리가 가리키는 데이터 블록이 카피-온-라이트(COW)가 수행되었는지 판단하기 위해 상기 매핑 엔트리의 처음 할당 비트(FAB)와 스냅샷 상태 비트(SSB)를 사용하여 처리하는 단계;

카피-온-라이트(COW)가 수행된 경우 FAB나 SSB를 초기화하고, 상기 매핑 블록의 변경을 디스크에 반영하는 단계;

상기 매핑 블록의 잠금을 해제하는 단계; 및

모든 매핑 블록에 대한 초기화가 완료되면 스냅샷 볼륨을 삭제하는 단계를 포함하는 것을 특징으로 하는 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법.

【청구항 6】

스토리지 에리어 네트워크(SAN) 기반의 네트워크 스토리지에서 볼륨의 데이터 블록에 대한 쓰기 연산을 수행하는 방법에 있어서,

스냅샷이 존재하는지 아닌지를 판단하여 매핑을 수행하는 단계;

현재 쓰기 연산의 대상이 되는 논리 데이터 블록에 대한 매핑 엔트리가 존재하는 매핑 블록 및 매핑 엔트리의 위치를 찾는 단계;

상기 매핑 블록을 디스크로부터 읽어서 원하는 매핑 엔트리의 값을 구하는 단계;

상기 매핑 엔트리의 처음 할당 비트(FAB) 값을 확인하여 데이터 블록이 스냅샷 생성 이후 처음 할당, 사용되는지 판단하는 단계;

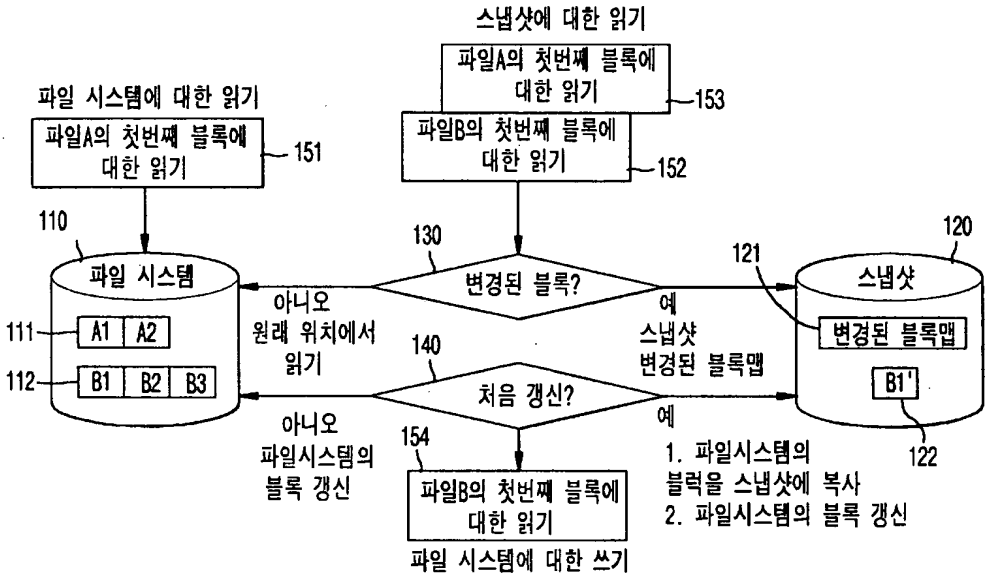
FAB 값이 0이고 매핑 엔트리 값이 초기값인 경우, 새로운 블록을 할당하고 데이터 내용을 복사 디스크에 기록하고, FAB 값을 1로 변경하고 원본 매핑 블록을 디스크에 반영하는 단계;

데이터 블록이 스냅샷 이전에 할당된 경우 스냅샷 상태 비트(SSB) 값을 통하여 COW가 수행되었는지를 판단하는 단계; 및

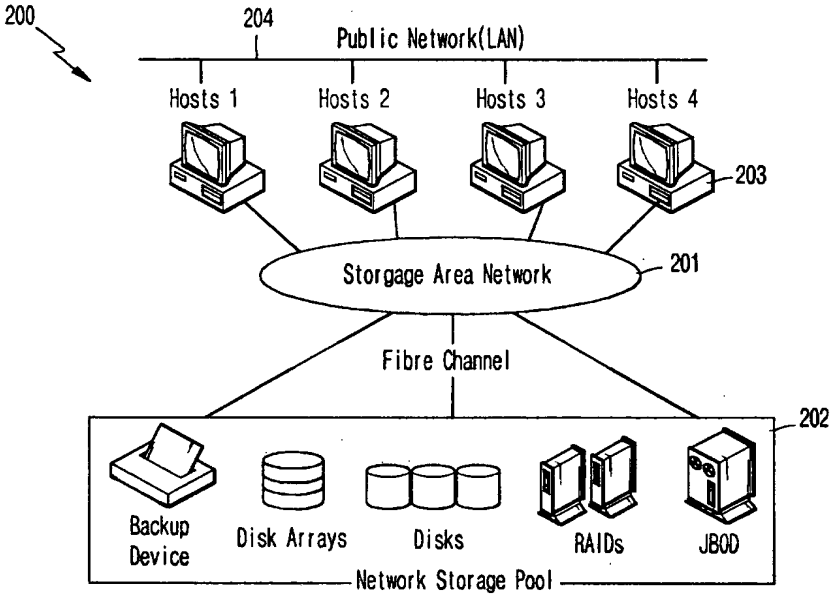
SSB가 0인 COW가 수행되지 않은 경우 COW를 수행하고 현재 스냅샷에 대한 SSB 값을 1로 변경하고, 디스크에 원본 매핑 블록을 기록하는 단계를 포함하는 것을 특징으로 하는 대용량 공유 저장장치를 위한 효율적인 스냅샷 수행방법.

【도면】

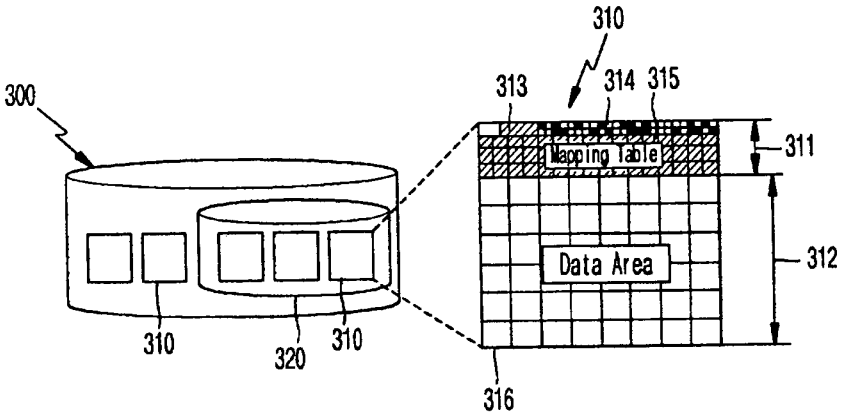
【도 1】


스냅샷에 대한 읽기
파일A의 첫번째 블록에 대한 읽기
153
파일B의 첫번째 블록에 대한 읽기
152
파일 시스템에 대한 읽기
파일A의 첫번째 블록에 대한 읽기
151
110
파일 시스템
130
변경된 블록?
121
120
스냅샷
111
A1 A2
112
B1 B2 B3
아니오
원래 위치에서 읽기
예
스냅샷 변경된 블록맵
변경된 블록맵
B1'
140
처음 갱신?
아니오
파일시스템의 블록 갱신
예
154
1. 파일시스템의 블럭을 스냅샷에 복사
2. 파일시스템의 블록 갱신
122
파일B의 첫번째 블록에 대한 읽기
파일 시스템에 대한 쓰기

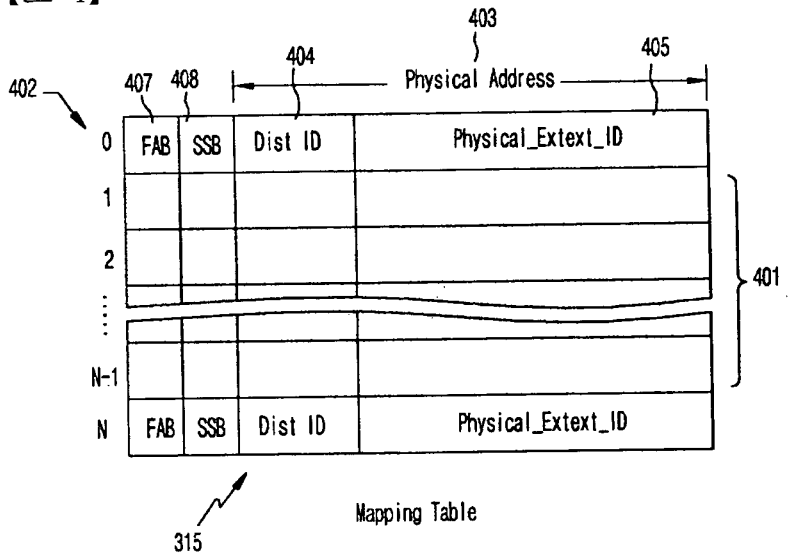
【도 2】


200
204
Public Network(LAN)
Hosts 1
Hosts 2
Hosts 3
Hosts 4
203
Storgage Area Network
201
Fibre Channel
202
Backup Device
Disk Arrays
Disks
RAIDs
JBOD
Network Storage Pool

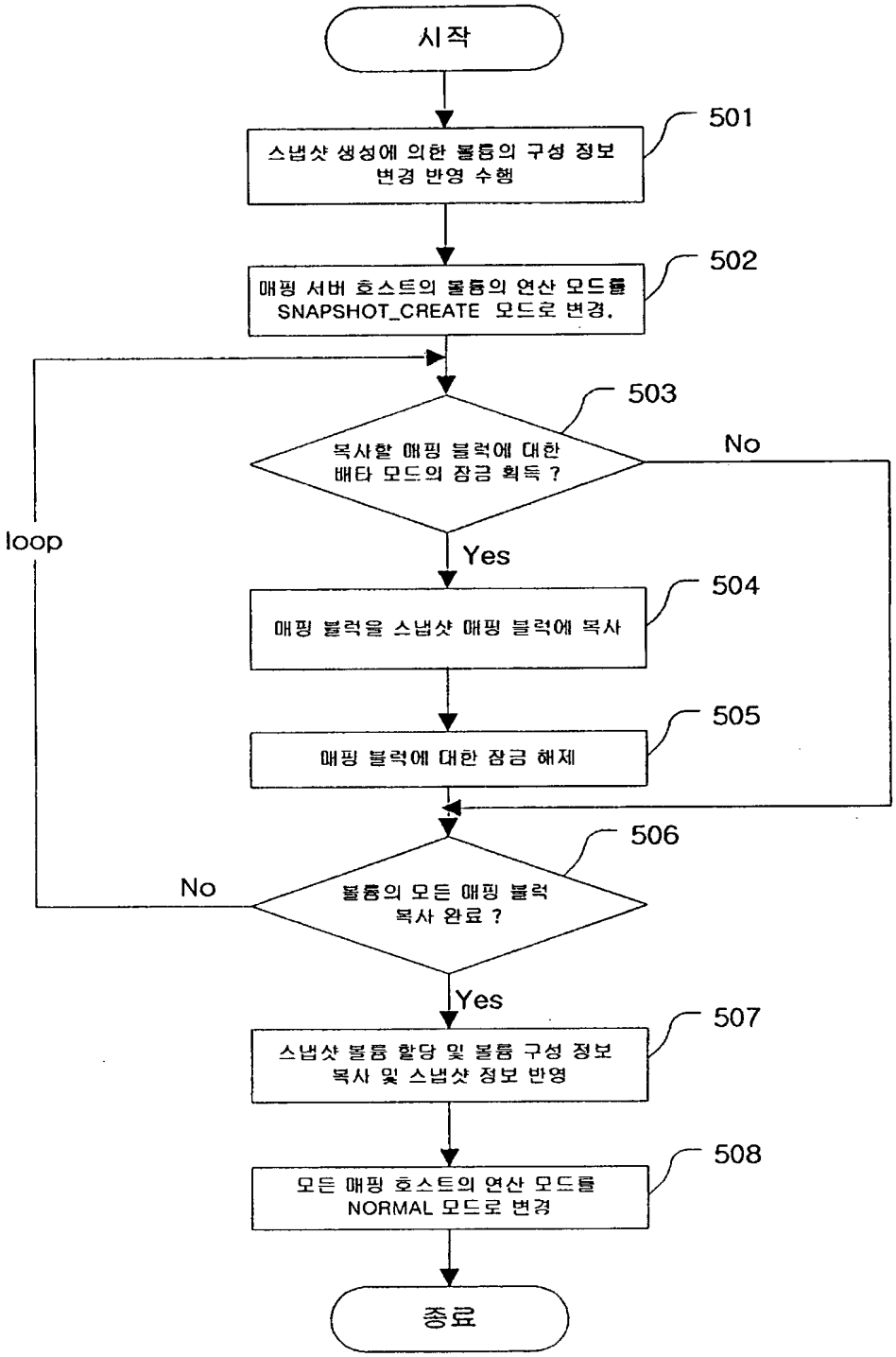
【도 3】


310
313
314
315
300
Mapping Table
311
Data Area
312
310
320
310
316

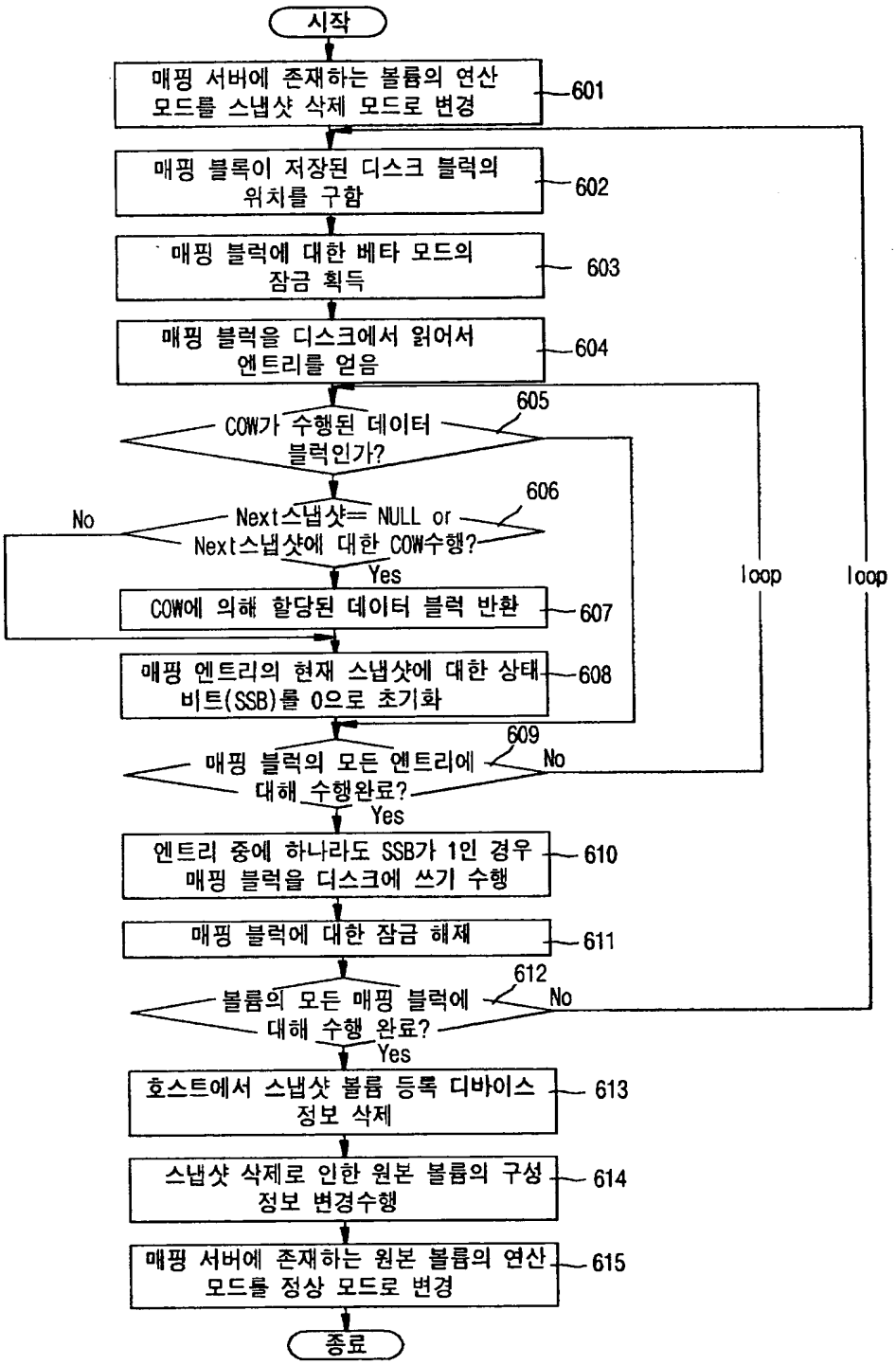
【도 4】


403
404
405
402
407
408
Physical Address
0
FAB
SSB
Dist ID
Physical_Extext_ID
1
2
401
N-1
N
FAB
SSB
Dist ID
Physical_Extext_ID
315
Mapping Table

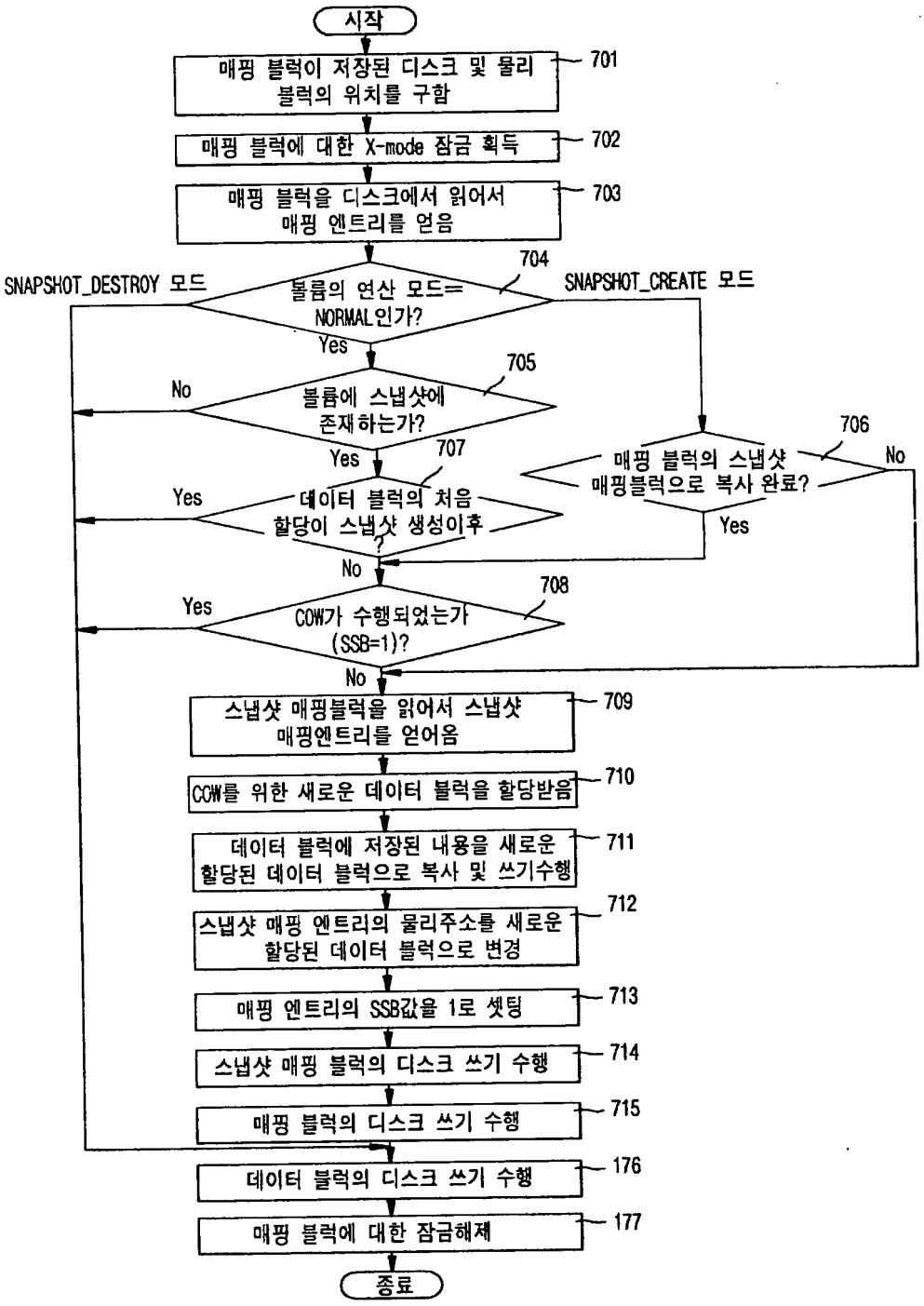
【도 5】


시작
501
스냅샷 생성에 의한 볼륨의 구성 정보 변경 반영 수행
502
매핑 서버 호스트의 볼륨의 연산 모드를 SNAPSHOT_CREATE 모드로 변경.
503
복사할 매핑 블럭에 대한 배타 모드의 잠금 획득 ?
No
Yes
loop
504
매핑 블럭을 스냅샷 매핑 블럭에 복사
505
매핑 블럭에 대한 잠금 해제
506
No
볼륨의 모든 매핑 블럭 복사 완료 ?
Yes
507
스냅샷 볼륨 할당 및 볼륨 구성 정보 복사 및 스냅샷 정보 반영
508
모든 매핑 호스트의 연산 모드를 NORMAL 모드로 변경
종료

【도 6】


시작
매핑 서버에 존재하는 볼륨의 연산 모드를 스냅샷 삭제 모드로 변경
601
매핑 블록이 저장된 디스크 블럭의 위치를 구함
602
매핑 블럭에 대한 베타 모드의 잠금 획득
603
매핑 블럭을 디스크에서 읽어서 엔트리를 얻음
604
605
COW가 수행된 데이터 블럭인가?
606
Next스냅샷= NULL or Next스냅샷에 대한 COW수행?
No
Yes
COW에 의해 할당된 데이터 블럭 반환
607
매핑 엔트리의 현재 스냅샷에 대한 상태 비트(SSB)를 0으로 초기화
608
609
매핑 블럭의 모든 엔트리에 대해 수행완료?
No
Yes
loop
loop
엔트리 중에 하나라도 SSB가 1인 경우 매핑 블럭을 디스크에 쓰기 수행
610
매핑 블럭에 대한 잠금 해제
611
612
볼륨의 모든 매핑 블럭에 대해 수행 완료?
No
Yes
호스트에서 스냅샷 볼륨 등록 디바이스 정보 삭제
613
스냅샷 삭제로 인한 원본 볼륨의 구성 정보 변경수행
614
매핑 서버에 존재하는 원본 볼륨의 연산 모드를 정상 모드로 변경
615
종료

【도 7】


시작
매핑 블럭이 저장된 디스크 및 물리 블럭의 위치를 구함
701
매핑 블럭에 대한 X-mode 잠금 획득
702
매핑 블럭을 디스크에서 읽어서 매핑 엔트리를 얻음
703
704
SNAPSHOT_DESTROY 모드
볼륨의 연산 모드= NORMAL인가?
SNAPSHOT_CREATE 모드
Yes
705
No
볼륨에 스냅샷에 존재하는가?
Yes
707
706
매핑 블럭의 스냅샷 매핑블럭으로 복사 완료?
No
Yes
Yes
데이터 블럭의 처음 할당이 스냅샷 생성이후?
No
708
Yes
COW가 수행되었는가 (SSB=1)?
No
스냅샷 매핑블럭을 읽어서 스냅샷 매핑엔트리를 얻어옴
709
COW를 위한 새로운 데이터 블럭을 할당받음
710
데이터 블럭에 저장된 내용을 새로운 할당된 데이터 블럭으로 복사 및 쓰기수행
711
스냅샷 매핑 엔트리의 물리주소를 새로운 할당된 데이터 블럭으로 변경
712
매핑 엔트리의 SSB값을 1로 셋팅
713
스냅샷 매핑 블럭의 디스크 쓰기 수행
714
매핑 블럭의 디스크 쓰기 수행
715
데이터 블럭의 디스크 쓰기 수행
176
매핑 블럭에 대한 잠금해제
177
종료